DOCUMENT RESUME

ED 111 420 IR 002 512

AUTHOR Heines, Jesse M.
TITLE Style and Communication in Interactive Programming.
PUB DATE Aug 75
NOTE 19p.; Paper presented at the Association for the Development of Computer-Based Instructional Systems Summer Conference (Portland, Maine, August 4-7, 1975) Not available in hard copy due to marginal reproducibility of original document

EDRS PRICE MF-$0.76 Plus Postage. HC Not Available from EDRS.
DESCRIPTORS Communication (Thought Transfer); Computer Programs; *Computers; Guidelines; Human Engineering; *Interaction; *Man Machine Systems; *Media Research; Programing

ABSTRACT

Research on man-machine communication was examined to gain insight into techniques for improving interactive programs through the enhancement of communicative style. The human-computer interaction is compared to a conversation, and specific recommendations for improving this interaction are enumerated. The suggestions are general in nature and are arranged into a preliminary "Guide to Style in Interactive Programing." A simple interactive program that tries to incorporate some of the recommendations cited is included as an appendix. (SK)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED111420

BEST COPY AVAILABLE

STYLE AND COMMUNICATION

IN

INTERACTIVE PROGRAMMING

Jesse M. Heines

Training Course Developer
Educational Services
Digital Equipment Corporation
Maynard, Massachusetts 01754

A paper presented at the

1975 Summer Conference
of the
Association for the Development of Computer-based
Instructional Systems

Portland, Maine
August 1975

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

IR 002 512

STYLE AND COMMUNICATION IN INTERACTIVE PROGRAMMING

Jesse M. Heines

ABSTRACT

Research on man-machine communication was examined to gain insight into techniques for improving interactive programs through the enhancement of communicative style. The human-computer interaction is compared to a conversation, and specific recommendations for improving this interaction are enumerated. The suggestions are general in nature and are arranged into a preliminary "Guide to Style in Interactive Programming". A sample interactive program that tries to incorporate some of the recommendations cited is included as an appendix.

INTRODUCTION

Few people who have interacted personally with a computer have lukewarm feelings about the experience. Most often, people find the interaction either highly enjoyable or totally distasteful (Martin, 1973; Melnyk, 1972). The strengths of these reactions are an important consideration in the design of computer-assisted instructional (CAI) programs, as students who find the interaction distasteful will be reluctant to use the computer repeatedly.

It is this author's belief that most of the distasteful qualities of interactive computer programs may be attributed directly to their poor communicative style. This is because program authors often fail to consider the posture of the naive user. This paper attempts to provide guidlines for good communicative style by examining the research on man-machine communication and techniques that can be used to smooth the human-computer interface.

COMPUTER INTERACTION AS A CONVERSATION

Nickerson (1969) suggests that little work in the related fields of ergonomics, human factors, and human engineering can be applied directly to the design of effective human-computer interactions. "What makes the man-computer interaction qualitatively different from other types of man-machine interactions", he explains, "is the fact that (man-computer interaction) may be described, without gross misuse of words, as a conversation". This theory can be supported by comparing Schramm's model of communication (1954), shown in Figure 1, to a diagram of the processes involved in human-computer interactions, shown in Figure 2.

From these diagrams, it might appear that the ideal human-computer interaction would be an exact replica of a human-human conversation. Chapinis (1971) and Foley (1973) point out several reasons why this is not yet technologically possible, and even a brief examination of this approach will show why it is not desirable in many applications. For example, graphic display techniques can impart far more information than verbal channels (Martin, 1973), and interactions with computers can improve upon normal technical conversations by reducing redundancy (Nickerson, 1969). The most desirable type of human-computer interaction, then, might be described as the one which allows the most efficient operation of the human-computer system. That is, it should be designed to provide the easiest-to-use interface between the problem defined by the user and the related capabilities of the computer (Foley, 1973; Melnyk, 1972).

In our struggle for efficiency, however, we often sacrifice the convenience of the user for the convenience of the system. Users are forced to understand cryptic messages and respond with codes rather than with words. Somehow, we forget what it is like to be users the minute we become programmers, just as we forget what it is like to be pedestrians when we become drivers. Our programs then impart the character of a cold automaton rather than a human author.

The recommendations presented in the remainder of this paper are intended to provide preliminary guidelines for programmers who wish to rehumanize their programs but who have only a minimum computer interface (standard teletypewriter or small cathode-ray tube) and knowledge of a high level language such as BASIC. They are interpretations (both inductive and deductive) of related literature and should not be construed as recommendations specifically intended by the referenced authors in their original contexts.

A PRELIMINARY GUIDE TO STYLE IN INTERACTIVE PROGRAMMING

> As yet, no acknowledged sense of style has developed for CAI... In the meantime, however, some singularly unstylish CAI programs are being written. (Martin, 1973, p. 413)

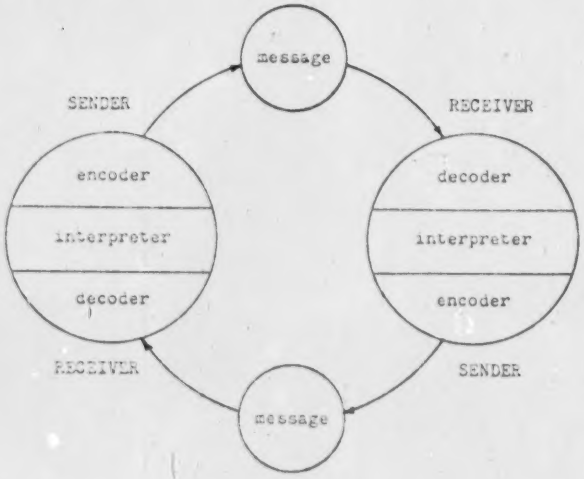

Figure 1

Schramm's Model of Communication

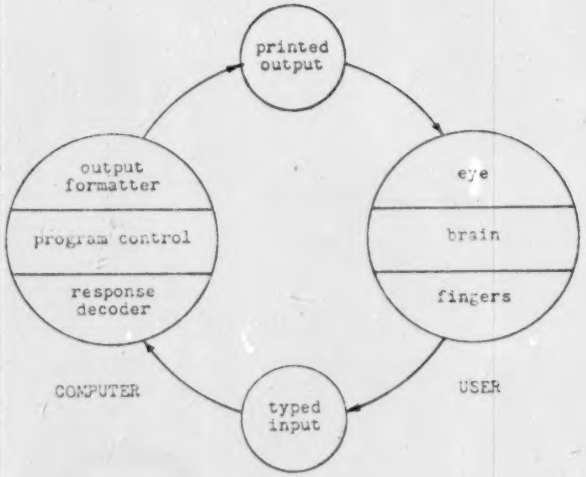

Figure 2

Human/Computer Interaction

(1) Maximize the amount of interaction in your programs. Meridith (1973) suggests that "machine surrogate tutors" best impart information through continuous interaction. Foley (1973) points out that a program will have the best chance of understanding the user's desires if it can get the user to supply a "stream of input". Yntema (1969) has observed that as interactions become more "expensive" (both in the monetary sense and in the number allowed), computer users are far more anxious about making errors. Interaction can be maximized by keeping your messages short and requiring a user response after every few lines.

(2) Tie your programs in with other media. Sometimes, short messages do not provide enough latitude to tell the user all that is necessary. But rather than print out several pages of text on the terminal, Heines (1975) suggests that a user's guide be written to accompany the program. The user's guide might include diagrams and photographs which do not lend themselves to computer display or just descriptions that are easier to read from a printed page than from the computer terminal.

(3) Use upper and lower case if available. This recommendation has been made by Repko (1975), among others, and is consistent with her view that a computer system should "conform to the user's conception of the environment". As text is normally presented in upper and lower case, so should it be on the computer terminal (if physically possible). Upper case text has a cold, official feeling while lower case text is less forbidding.

(4) Display program output along the entire width of your screen or paper. Gregory and Poulton (1970) found that poor readers had significantly poorer reading comprehension when text of seven words per line was right-justified as compared to their ability to comprehend the same material with uneven right-hand margins. (Good readers showed no significant difference in comprehension with the two methods of presentation.) This effect was nullified, however, when the text was lengthened to twelve words per line. The primary implication of this finding is that text of seven words per line or less should not be right-justified. The secondary implication is that text lines should be made as long as possible, at least up to twelve words per line. Thus the width of the output device (screen or paper) should limit the length of your message lines rather than the convenience of the program.

For example, consider the normal use of the PRINT statement to produce program output in BASIC. If the length of a statement line is limited to the width of the output medium, program output must be at least eight characters shorter than the width of the medium. This is because at least one space is needed for the line number, five for the PRINT command, and two for the opening and closing quotes. The problem can be easily solved by using a semicolon at the end of a PRINT statement and continuing the additional text with the next statement line.

(5) Keep format and style in mind. McLaughlin (1966) compared the abilities of college undergraduates to locate information in well-produced and poorly produced (verbose) technical pamphlets. He found no significant difference in test performance when the two types of pamphlets were used by motivated students. Unmotivated students, however, showed significantly poorer performance when they used the poorly produced pamphlet as compared to their performance with the well-produced one. In both cases -- motivated and

unmotivated -- students spontaneously stated that they would not have read the poorer version voluntarily. McLaughlin concludes:

> Objective measurement may show that the style of presentation of printed technical matter has little effect upon the efficiency with which information can be culled from it. Yet subjective preferences may be so strong as to make readers ignore material presented in a certain style. (Page 257)

(6) Consider the experience of your target population. Mills (1967) and Nickerson (1969) have stated that the population of computer users is becoming increasingly heterogeneous. This means that more and more naive users are continually coming into contact with interactive computer programs. The style of these programs should therefore be friendly and conversational (Martin, 1973). Repko (1975) feels that programs written for naive users must not assume the programmer's knowledge of computer terms and operations. She acknowledges the difficulty of "putting yourself into your user's shoes" by describing the programmer as seeing the computer from the "inside" while the user sees it from the "outside".

Consider the act of entering data to a program in an interactive mode. In BASIC, the program statement used for this purpose is INPUT, which prints a question mark on the terminal and accepts data typed at the keyboard. Very often, therefore, one sees interactive programs which print out queries like this:

```
INPUT THE INTEREST RATE IN % PER YEAR
?
```

This query clearly reflects the programmer's view of the data entry procedure. The user must interpret the word "input" as "type". With little additional effort, the programmer can use the question mark as normal punctuation and relate more closely to the user's view of the data entry procedure:

```
WHAT IS THE INTEREST RATE IN % PER YEAR?
```

This is a simple question, and the fact that a user response in required is more obvious.

(7) Prompt the user as to the type of response required. Even the most obvious query to an interactive programmer may not immediately indicate to a naive user the type of response to be made. When users are prompted, however, the doubt is quickly erased (Heines, 1974). For example,

```
PLEASE TYPE "YES" OR "NO" IN RESPONSE TO THE FOLLOWING
QUESTION AND THEN PRESS THE RETURN KEY:

HAVE YOU EVER USED THIS TERMINAL BEFORE?
```

Once this type of instruction is given, shorter prompts usually suffice:

```
WOULD YOU LIKE TO RUN THE PROGRAM AGAIN NOW ("YES" OR "NO")?
```

(8) Use menus to indicate the user's options. Option menus have been used successfully with all classes of computer users (Foley, 1973; Martin, 1973). This technique allows the user to select an option from a given list quickly and efficiently because all available options are displayed and indication of the option desired by the user is extremely simple. Following is an example of a simple option menu which guides the student through an interactive environment (Heines, 1974):

> YOU ARE NOW REGISTERED FOR THIS TERMINAL SESSION AND MAY SELECT A PROGRAM OPTION FROM THE FOLLOWING LIST:
>
> (1) RUN A CHECK POINT PROGRAM
> (2) DISPLAY ALL THE DATA STORED ON YOUR WORK
> (3) DISPLAY ALL STORED DATA IN SUMMARY FORM
> OR...
> (4) END THIS TERMINAL SESSION
>
> WHICH OPTION WOULD YOU LIKE TO EXECUTE NOW (TYPE A NUMBER)?

(9) Make error messages friendly and factual. Programmers very often overlook the importance of carefully constructed error messages. In the worse case, their error messages are flippant insults to the user. At best, error messages are often omitted, queries answered incorrectly are simply reprinted, and the user is surprised to see a question asked again that he or she has just answered. Consider the following interaction:

> Computer: HAVE YOU EVER USED THIS TERMINAL BEFORE?
> User : YSE
> Computer: HAVE YOU EVER USED THIS TERMINAL BEFORE?

The naive user will surely be confused, thinking that he or she has already responded "YES" to this query.

One solution is to tell the user that an incorrect response has been made. When this is done, however, Meredith (1973) suggests that messages using terms such as "not understood" or "rephrase" should be used rather than "that most irritating word in the programmer's lexicon: ILLEGAL!"

Foley (1973) stresses that "to the user, each error is a unique obstacle". The programmer must handle unanticipated responses "elegantly", he continues, encouraging the user to correct the error. He notes:

> A reference librarian is very unlikely to tell a user that she has no idea what he is talking about -- yet this is exactly what computer information systems regularly do. Thus they quickly gain a reputation for being frustrating.

The following example of error handling improves upon the interaction shown previously:

```
Computer:  HAVE YOU EVER USED THIS TERMINAL BEFORE?
User    :  YSE
Computer:  I CAN ONLY RECOGNIZE THE RESPONSES "YES"
           OR "NO" TO THIS QUESTION.  PLEASE CHECK
           YOUR RESPONSE AND TRY AGAIN.

           HAVE YOU EVER USED THIS TERMINAL BEFORE?
```

This type of explanatory error message can be sufficiently generalized to be programmed as a subroutine and called whenever a "yes" or "no" response is required.

(10) Do not eliminate message redundancy at the expense of message clarity. Some readers may feel that the above message is far too wordy to be practical, especially if it is repeated each time this mistake is made. But the balance of message redundancy and clarity is often delicate: too much redundancy can bore an audience while too little can confuse them (Schramm, 1954). Nickerson (1969) admits that "all users tend to be impatient with redundant and non-informative messages", but further notes that:

> ...the extent to which any particular communication from the computer is redundant or non-informative depends upon the amount of experience that the user has had with the system.

For example, the message:

```
DA 90
```

may be sufficient for some users but non-informative for others. The message:

```
OUT OF DATA AT LINE 90
```

yields more information, but may be redundant for experienced users.

A solution suggested by Nickerson (1969) is to use shorter abbreviations and mnemonics, but to allow the user to view the longer, less cryptic message by entering, for example, "what" or "?". One should also consider the display rate of the user's terminal when planning error messages, as longer messages are tolerable when they are displayed quickly but intolerable if they are displayed slowly.

(11) Give the user as much feedback as possible. Foley (1973) and Schramm (1954) have pointed out the importance of feedback for the successful operation of any communication system. Melnyk (1972) and Meredith (1973) have related the use of feedback to interactive computer programs in the form of error messages for incorrect input. But feedback can also keep the user informed of the state of the system. For example, naive users are often confused when the terminal pauses if input is not required, as may be the case while a tape or disk file is being processed. Confusion can be avoided by printing, for example:

```
YOJR SCORE IS NOW BEING RECORDED...
```

or, more simply:

ONE MOMENT, PLEASE...

(12) Use graphics wherever possible. Graphics need not be limited to expensive, sophisticated systems. Even simple diagrams can be very helpful in trying to communicate ideas. While teletypewriters are extremely slow for displaying graphics, small cathode-ray tubes (CRT's) usually have special functions such as tab, backspace, and screen clear which can be used to speed up the rate at which graphics can be displayed. These features make it feasible to draw graphs and diagrams at a reasonable rate within the limitations of the terminal's character set.

(13) Write your programs so that they can be changed, improved, and enhanced. As interactive programs are used, their strengths and weaknesses become apparent. If programmed in a haphazard manner, their weaknesses can be very difficult to correct, even if they involve only a small change in wording and structure. Repko (1975) suggests that programs be made flexible by functional division, isolating the input and output sections (see Figure 3). She comments:

> The mechanism of the program should never be an excuse for not allowing changes in the man-machine communication.

The simplest way to make programs adaptable is to document them extensively. The reader is referred to the program listing in the appendix as an example of a BASIC language program that was extensively documented by using the REMARK statement.

(14) Never give in to the machine. Anyone who has asked a busy programmer for assistance on a programming problem has heard the reply, "It can't be done". Usually, this simply means that the problem appears to be non-trivial and the programmer does not wish to take the time to help you. Time and again, however, computer people have proven that there is some truth in the saying, "the impossible we do immediately; miracles take a little longer". No matter how impossible your idea might seem at first, you can usually implement it in some form even if you have to compromise slightly. Careful scrutiny of your system will almost always reveal ways to get around limitations imposed by the hardware and software.

CONCLUSION

The guidelines presented in this paper are not novel. When viewed in retrospect, most of the recommendations appear to be the product of plain common sense. It is encouraging to note, however, that actual research does support these simple ideas. It is this author's hope that further research will expand this effort and that interactive programs written with a clear communicative style will make it easier for people from all backgrounds to use the computer effectively and enjoyably.

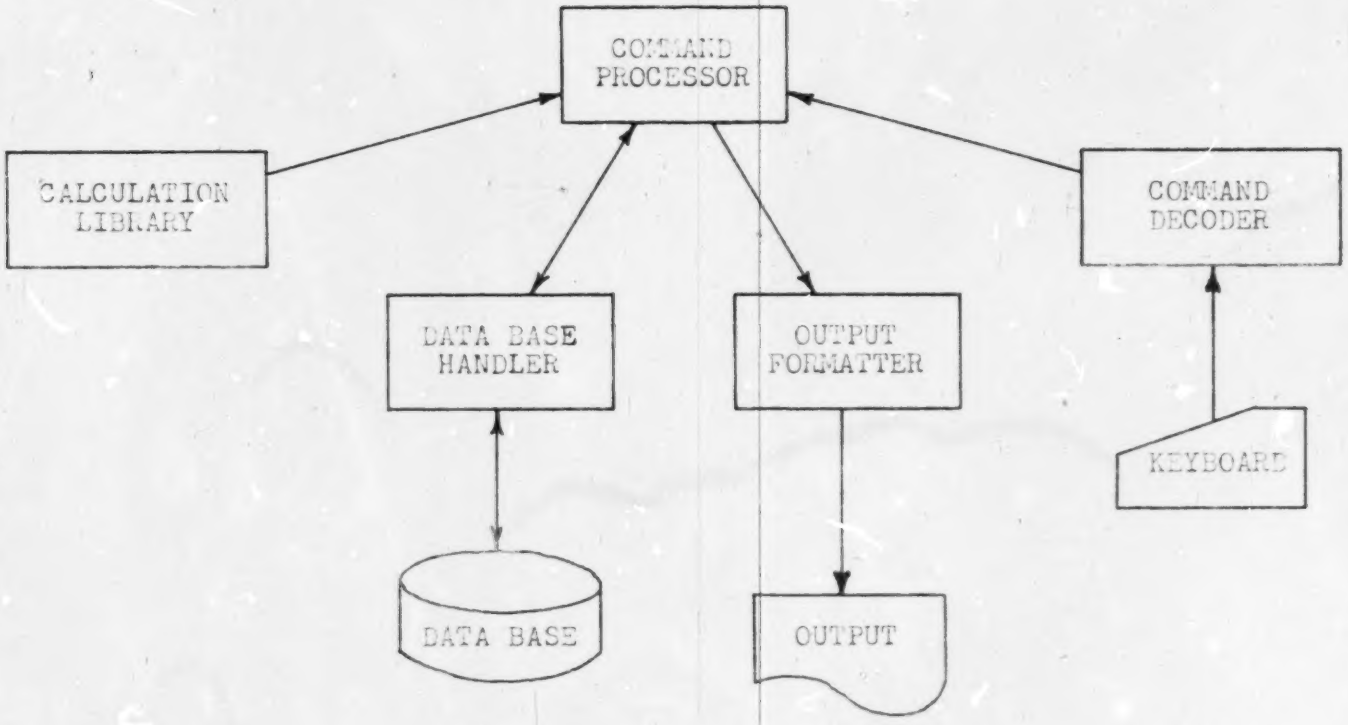

Figure 3

Functional Division of a Computer Program

[adapted from Repko (1975), page 7]

REFERENCES CITED

Chapinis, Alphonse. Prelude to 2001: explorations in human communication. American Psychologist 26(11):949-961, November 1971.

Foley, Joseph M. Communication aspects of information science. Theory Into Practice 12(3):167-172, June 1973.

Gregory, Margaret and E.C. Poulton. Even versus uneven right-hand margins and the rate of comprehension in reading. Ergonomics 13(4):427-434, July 1970.

Hermes, Jesse M. An Interactive, Computer-Managed Model for the Evaluation of Audio-Tutorial Instruction. Unpublished Master's Thesis, University of Maine at Orono, May 1974.

_____. Everything you always wanted to know about computers ... you can teach yourself. Educational Technology 15(5):47-50, May 1975.

Martin, James. Design of Man-Computer Dialogues. Prentice-Hall, Inc., Englewood Cliffs, N.J. 1973, pp. 373-423.

McLaughlin, G. Harry. Comparing styles of presenting technical information. Ergonomics 9(3):257-259, May 1966.

Melnyk, Vera. Man-machine interface: frustration. Journal of the American Society for Information Science 23(6):392-401, November-December 1972.

Meredith, J.C. Machine as tutor. The Computer and Education. (The Educational Technology Review Series: Number Nine.) Educational Technology Publications, Englewood Cliffs, N.J. 1973, pp. 27-34.

Mills, R.G. Man-machine communication and problem solving. In C.A. Cuadra (ed.), Annual Review of Information Science and Technology, Volume 2. Interscience Publications, N.Y. 1967.

Nickerson, R.S. Man-computer interaction: a challenge for human factors research. Ergonomics 12(4):501-517, July 1969.

Repko, Marya. Man-machine communication. Decuscope 14(1):5-7, February 1975.

Schramm, Wilbur. How communication works. In Wilbur Schramm (ed.), The Process and Effects of Mass Communication. University of Illinois Press, Urbana, Illinois. 1954, pp. 3-10.

Yntema, D.B. Engineering psychology in man-computer interaction. Paper presented at the Annual Convention of the American Psychological Association, San Francisco, California. 1969.

APPENDIX: PROGRAM EXAMPLE

A total of fourteen recommendations for the enhancement of interactive program style were given in this paper. These guidelines were used by the author to modify and expand a program originally written by Bob Albrecht (People's Computer Company, Menlo Park, California). This program, originally called HURKLE, was design to give students practice in working with the Cartesian coordinate system in a game format.

A sample run of the modified program, called HURKØ2, appears on the following two pages. This print-out was photocopied directly from a computer terminal, but the input supplied by the user has been underlined for clarity. The numbers typed in the right-hand margin correspond to the numbers of the specific recommendations in the body of this paper that were considered when that part of the program was designed. A listing of the BASIC language statements that make up this program (in OS/8 BASIC) appears on the four pages following the program print-out.

RUNNH

HURKLE TWO

```
DO YOU WISH TO SEE THE INSTRUCTIONS ("YES" OR "NO") ?YES                    ( 1)

A HURKLE IS HIDING IN A CARTESIAN COORDINATE GRID                           ( 2)
LIKE THE ONE AT THE RIGHT.  GUESS ITS LOCATION BY          (POS)            ( 5)
ENTERING A HORIZONTAL COORDINATE FOLLOWED BY A               N              (12)
VERTICAL ONE.  FOR EXAMPLE, THE  *  IS AT  -4,1 ..        *  !
POINT  0,0  IS AT THE CENTER OF THE GRID, WHERE          W <--+--> E
THE  +  IS.  AFTER EACH GUESS, I WILL TELL YOU        (NEG)  !  (POS)
WHERE TO GO TO FIND THE HURKLE BY SAYING "NORTH"             S
FOR THE POSITIVE VERTICAL DIRECTION, "WEST" FOR           (NEG)
THE NEGATIVE HORIZONTAL, ETC.  GOOD LUCK!

YOUR AVAILABLE OPTIONS ARE NOW "GO", "HELP", "INSTR", "QUIT", "SIZE",       ( 8)
AND "TRIES".  WHICH WOULD YOU LIKE TO EXERCISE (ENTER A WORD) ?GO

THE HURKLE IS HIDING IN A 10 BY 6 COORDINATE GRID.  HORIZONTAL              ( 6)
VALUES GO FROM -5 TO 5 AND VERTICAL VALUES GO FROM -3 TO 3 .  FIND
THE HURKLE WITHIN 6 GUESSES!

YOUR FIRST GUESS (ENTER COORDINATES SEPARATED BY A COMMA) ?0,0              ( 7)
GO SOUTHWEST...

YOUR SECOND GUESS ?-4,-4
   YOUR SECOND COORDINATE IS OUTSIDE OF THE HURKLE'S GRID.  TRY AGAIN...    ( 9)

YOUR SECOND GUESS ?-4,-2
GO SOUTHEAST...

YOUR THIRD GUESS ?-2,-3
GO EAST...

YOUR FOURTH GUESS ?-1,-3

HURK!  HURK!  YOU FOUND THE HURKLE IN 4 GUESSES!!

IF YOU'D LIKE TO PLAY AGAIN, PLEASE ENTER THE "GO" OPTION BELOW.

YOUR AVAILABLE OPTIONS ARE NOW "GO", "HELP", "INSTR", "QUIT", "SIZE",
AND "TRIES".  WHICH WOULD YOU LIKE TO EXERCISE (ENTER A WORD) ?HELP         ( 7)

YOUR OPTIONS PERFORM THE FOLLOWING FUNCTIONS:
        GO      LOCATE THE HURKLE AT A NEW GRID POINT AND ALLOW YOU TO      ( 8)
                   GUESS WHERE IT IS HIDING
        HELP    DISPLAY THIS MESSAGE
        INSTR   DISPLAY THE INSTRUCTIONS
        QUIT    END THE GAME
        SIZE    CHANGE THE SIZE OF THE GRID IN WHICH THE HURKLE CAN HIDE
        TRIES   CHANGE THE NUMBER OF TRIES ALLOWED TO FIND THE HURKLE
TO MAKE THE COMPUTER EXERCISE AN OPTION, SIMPLY TYPE ITS KEYWORD BELOW.

YOUR AVAILABLE OPTIONS ARE NOW "GO", "HELP", "INSTR", "QUIT", "SIZE",      (10)
AND "TRIES".  WHICH WOULD YOU LIKE TO EXERCISE (ENTER A WORD) ?SIZE
```

```
THE CURRENT SIZE OF THE HURKLE'S GRID IS 10 BY 6 (HORIZONTAL BY
VERTICAL).  YOUR NEW DIMENSIONS MUST ALSO BE EVEN INTEGERS.  ENTER YOUR
NEW DIMENSIONS BELOW SEPARATED BY A COMMA; HORIZONTAL DIMENSION FOLLOWED   ( 4)
BY VERTICAL.  YOU MAY LEAVE THE GRID SIZE UNCHANGED BY ENTERING "0,0".

YOUR NEW DIMENSIONS ?20,,20
   YOUR SECOND DIMENSION IS NOT AN EVEN INTEGER.  PLEASE TRY AGAIN...      ( 9)

YOUR NEW DIMENSIONS ?20,20

THE NEW SIZE OF THE HURKLE'S GRID IS 20 BY 20 .                           (11)


YOUR AVAILABLE OPTIONS ARE NOW "GO", "HELP", "INSTR", "QUIT", "SIZE",
AND "TRIES".  WHICH WOULD YOU LIKE TO EXERCISE (ENTER A WORD) ?TRIES


YOU ARE NOW ALLOWED 6 TRIES TO FIND THE HURKLE.  ENTER YOUR NEW LIMIT
BELOW.  YOU MAY LEAVE THE LIMIT UNCHANGED BY ENTERING "0".

YOUR NEW LIMIT ?4

YOU WILL NOW BE ALLOWED 4 TRIES TO FIND THE HURKLE.                       (11)


YOUR AVAILABLE OPTIONS ARE NOW "GO", "HELP", "INSTR", "QUIT", "SIZE",
AND "TRIES".  WHICH WOULD YOU LIKE TO EXERCISE (ENTER A WORD) ?GO


THE HURKLE IS HIDING IN A 20 BY 20 COORDINATE GRID.  HORIZONTAL
VALUES GO FROM -10 TO 10 AND VERTICAL VALUES GO FROM -10 TO 10 .  FIND
THE HURKLE WITHIN 4 GUESSES!


YOUR FIRST GUESS (ENTER COORDINATES SEPARATED BY A COMMA) ?0,0
GO SOUTH...

YOUR SECOND GUESS ?0,-5
GO SOUTH...                                                               ( 1)

YOUR THIRD GUESS ?0,-7
GO SOUTH...

YOUR FOURTH GUESS ?0,-8

SORRY, BUT YOU HAVE HAD THE LIMIT OF 4 GUESSES.  THE HURKLE WAS HIDING
AT POINT  0,-10 .

IF YOU'D LIKE TO PLAY AGAIN, PLEASE ENTER THE "GO" OPTION BELOW.          (10)


YOUR AVAILABLE OPTIONS ARE NOW "GO", "HELP", "INSTR", "QUIT", "SIZE",
AND "TRIES".  WHICH WOULD YOU LIKE TO EXERCISE (ENTER A WORD) ?QUIT


YOU PLAYED A TOTAL OF 2 GAMES AND FOUND THE HURKLE IN 1 OF THEM.          (11)
THAT'S A WINNING PERCENTAGE OF 50 % !

BYE!
```

```
LIST

HURK02  BA    3 0

1000 REM                * * * * * * * * * * * * * * * * * * *
1010 REM                *                                   *
1020 REM                *      H U R K L E     T W O        *
1030 REM                *                                   *
1040 REM                * * * * * * * * * * * * * * * * * * *
1050 REM
1060 REM
1070 REM
1080 REM  ORIGINAL "HURKLE" AUTHOR      BOB ALBRECHT
1090 REM                                PEOPLE'S COMPUTER COMPANY
1100 REM                                MENLO PARK, CALIFORNIA
1110 REM
1120 REM  ORGINALLY PUBLISHED IN        "101 BASIC GAMES"
1130 REM                                DIGITAL EQUIPMENT CORPORATION
1140 REM
1150 REM  ADAPTED FOR CLASSIC BY        JESSE M. HEINES
1160 REM                                DIGITAL EQUIPMENT CORPORATION
1170 REM
1180 REM                                JANUARY, 1975
1190 REM
1200 REM
1210 REM    * * * * * * *   V A R I A B L E   D I R E C T O R Y
1220 REM
1230 REM     VARIABLE        USAGE
1240 REM     --------        -----
1250 REM
1260 REM        A       INPUT CODE   0="NO", 1="YES"
1270 REM        A$      GENERAL ALPHANERIC USER INPUT
1280 REM        C0$     CHR$(34)  ["]
1290 REM        C1$     C0$ & ", " & C0$  [", "]
1300 REM        G1      HORIZONTAL GRID DIMENSION
1310 REM        G2      VERTICAL GRID DIMENSION
1320 REM        G3      USER-REQUESTED HORIZONTAL GRID DIMENSION
1330 REM        G4      USER-REQUESTED VERTICAL GRID DIMENSION
1340 REM        H       USER INPUT, HORIZONTAL GUESS
1350 REM        I1      TOTAL NUMBER OF GAMES PLAYED
1360 REM        I2      NUMBER OF GAMES IN WHICH THE HURKLE WAS FOUND
1370 REM        I3      NUMBER OF GUESSES IN GAMES COUNTED IN "I2"
1380 REM        K       GENERAL FOR-NEXT LOOP INDEX
1390 REM        N       NUMBER OF TRIES ALLOWED PER HIDING PLACE
1400 REM        N1      USER-REQUESTED NUMBER OF TRIES PER HIDING PLACE
1410 REM        F0      NUMERIC ARGUMENT PASSED TO A SUBROUTINE
1420 REM        T       GUESS COUNTER
1430 REM        T$(K)   ORDINAL EXPRESSION OF GUESS NUMBER
1440 REM        V       USER INPUT, VERTICAL GUESS
1450 REM        X       HORIZONTAL COORDINATE OF HURKLE'S HIDING PLACE
1460 REM        Y       VERTICAL COORDINATE OF HURKLE'S HIDING PLACE
1470 REM
1480 REM
1490 REM    * * * * * * *   D E C L A R A T I O N S
1500 REM
1510 LET C0$=CHR$(34)
1520 LET C1$=C0$ & ", " & C0$
1530 DIM T$(10,8)
1540 REM -- FOR DECSYSTEM 10, REPLACE ABOVE STATEMENT WITH
1550 REM        DIM T$(10)
1560 DATA "FIRST", "SECOND", "THIRD", "FOURTH", "FIFTH"
1570 DATA "SIXTH", "SEVENTH", "EIGHTH", "NINTH", "TENTH"
1580 FOR K=1 TO 10
1590 READ T$(K)
1600 NEXT K
1610 LET I1=0
1620 LET I2=0
1630 REM
1640 REM
1650 REM                        -------------------------
1660 REM    * * * * * * *       M A I N   P R O G R A M
1670 REM                        -------------------------
1680 REM
1690 REM
1700 GOSUB 3820
1710 PRINT "HURKLE TWO"
1720 PRINT "----------"
1730 PRINT
1740 PRINT
1750 PRINT "DO YOU WISH TO SEE THE INSTRUCTIONS (", C0$, "YES", C0$,
1760 PRINT " OR ", C0$, "NO", C0$, ")";
1770 RANDOMIZE
1780 LET G1=6+2*INT(4*RND(0))
1790 LET G2=6+2*INT(4*RND(0))
1800 LET N=5+INT(3*RND(0))
1810 GOSUB 3870
1820 IF A=0 THEN 1870
1830 GOSUB 4170
```

```
1840 REM
1850 REM    *******   O P T I O N   I N P U T
1860 REM
1870 PRINT
1880 PRINT "YOUR AVAILABLE OPTIONS ARE NOW ";
1890 P0=1
1900 GOSUB 4060
1910 PRINT "WHICH WOULD YOU LIKE TO EXERCISE (ENTER A WORD)";
1920 INPUT A$
1930 PRINT
1940 IF A$="GO" THEN 2090
1950 IF A$="HELP" THEN 2960
1960 IF A$="INSTR" THEN 1830
1970 IF A$="QUIT" THEN 4540
1980 IF A$="SIZE" THEN 3140
1990 IF A$="TRIES" THEN 3570
2000 PRINT "PLEASE ENTER ONLY ";
2010 P0=2
2020 GOSUB 4060
2030 PRINT "(";C0$;"HELP";C0$;" PRINTS AN EXPLANATION OF EACH.)  ";
2040 PRINT "YOUR CHOICE";
2050 GOTO 1920
2060 REM
2070 REM    *******   T H E   " G O "   O P T I O N
2080 REM
2090 REM
2100 REM  *******  SET THE HURKLE'S COORDINATES
2110 REM
2120 X=-G1/2+INT((G1+1)*RND(0))
2130 Y=-G2/2+INT((G2+1)*RND(0))
2140 PRINT C2$;
2150 GOSUB 3820
2160 PRINT "THE HURKLE IS HIDING IN A";
2170 IF SEG$(STR$(G1),LEN(STR$(G1)),LEN(STR$(G1)))<>"0" THEN 2210
2180 REM -- FOR DECSYSTEM-10, REPLACE ABOVE STATEMENT WITH
2190 REM         IF RIGHT$(STR$(G1),LEN(STR$(G1)))<>"0" THEN 2210
2200 PRINT "N";
2210 PRINT G1; "BY"; G2; "COORDINATE GRID   HORIZONTAL"
2220 PRINT "VALUES GO FROM "; G1/-2; "TO"; G1/2; "AND VERTICAL ";
2230 PRINT "VALUES GO FROM "; G2/-2; "TO"; G2/2; ".  FIND"
2240 PRINT "THE HURKLE WITHIN"; N; "GUESSES!"
2250 PRINT
2260 REM
2270 REM  *******  INPUT THE GUESSES
2280 REM
2290 FOR T=1 TO N
2300 PRINT
2310 IF T>10 THEN 2340
2320 PRINT "YOUR "; T$(T);
2330 GOTO 2350
2340 PRINT "YOUR"; T; "TH";
2350 PRINT " GUESS";
2360 IF T>1 THEN 2380
2370 PRINT " (ENTER COORDINATES SEPARATED BY A COMMA)";
2380 INPUT H,V
2390 REM
2400 REM  *******  CHECK GUESSES FOR VALIDITY
2410 REM
2420 IF H<-G1/2 THEN 2470
2430 IF H>+G1/2 THEN 2470
2440 IF V<-G2/2 THEN 2490
2450 IF V>+G2/2 THEN 2490
2460 GOTO 2550
2470 PRINT "   YOUR FIRST";
2480 GOTO 2500
2490 PRINT "   YOUR SECOND";
2500 PRINT " COORDINATE IS OUTSIDE OF THE HURKLE'S GRID!  TRY AGAIN..."
2510 GOTO 2300
2520 REM
2530 REM  *******  EVALUATE A VALID GUESS
2540 REM
2550 IF ABS(X-H)+ABS(Y-V)=0 THEN 2720
2560 IF N=T THEN 2620
2570 GOSUB 4380
2580 NEXT T
2590 REM
2600 REM  *******  OUT OF GUESSES
2610 REM
2620 PRINT
2630 PRINT
2640 PRINT "SORRY, BUT YOU HAVE HAD THE LIMIT OF"; N; "GUESSES.  THE ";
2650 PRINT "HURKLE WAS HIDING"
2660 PRINT "AT POINT  "; STR$(X); ","; STR$(Y); " ."
2670 P0=0
2680 GOTO 2810
2690 REM
2700 REM  *******  FOUND HURKLE MESSAGE
2710 REM
2720 PRINT
2730 IF T>5 THEN 2790
2740 FOR K=1 TO 6-T
2750 PRINT "HURK!  "; PNT(7);
2760 REM -- FOR DECSYSTEM-10, REPLACE ABOVE STATEMENT WITH
2770 REM         PRINT "HURK!  "; CHR$(7);
2780 NEXT K
```

```
2790 PRINT "YOU FOUND THE HURKLE IN"; T; "GUESSES!!"
2800 P0=1
2810 PRINT
2820 PRINT "IF YOU'D LIKE TO PLAY AGAIN, PLEASE ENTER THE "; C0$; "GO";
2830 PRINT C0$; " OPTION BELOW "
2840 PRINT
2850 REM*
2860 REM  ******* INCRENENT THE GAME AND TOTAL GUESSES COUNTERS
2870 REM
2880 LET I1=I1+1
2890 LET I2=I2+P0
2900 IF P0=0 THEN 1870
2910 LET I3=I3+T
2920 GOTO 1870
2930 REM
2940 REM    * * * * * * *   T H E   " H E L P "   O P T I O N
2950 REM
2960 GOSUB 3820
2970 PRINT "YOUR OPTIONS PERFORM THE FOLLOWING FUNCTIONS:"
2980 PRINT "   GO      LOCATE THE HURKLE AT A NEW GRID POINT AND ";
2990 PRINT "ALLOW YOU TO"
3000 PRINT "               GUESS WHERE IT IS HIDING"
3010 PRINT "   HELP    DISPLAY THIS MESSAGE"
3020 PRINT "   INSTR   DISPLAY THE INSTRUCTIONS"
3030 PRINT "   QUIT    END THE GAME"
3040 PRINT "   SIZE    CHANGE THE SIZE OF THE GRID IN WHICH THE ";
3050 PRINT "HURKLE CAN HIDE"
3060 PRINT "   TRIES   CHANGE THE NUMBER OF TRIES ALLOWED TO FIND ";
3070 PRINT "THE HURKLE"
3080 PRINT "TO MAKE THE COMPUTER EXERCISE AN OPTION, SIMPLY TYPE ";
3090 PRINT "ITS KEYWORD BELOW "
3100 GOTO 1870
3110 REM
3120 REM    * * * * * * *   T H E   " S I Z E "   O P T I O N
3130 REM
3140 GOSUB 3820
3150 PRINT "THE CURRENT SIZE OF THE HURKLE'S GRID IS"; G1; "BY"; G2;
3160 PRINT "(HORIZONTAL BY"
3170 PRINT "VERTICAL).  YOUR NEW DIMENSIONS MUST ALSO BE EVEN ";
3180 PRINT "INTEGERS.  ENTER YOUR"
3190 PRINT "NEW DIMENSIONS BELOW SEPARATED BY A COMMA, HORIZONTAL ";
3200 PRINT "DIMENSION FOLLOWED"
3210 PRINT "BY VERTICAL.  YOU MAY LEAVE THE GRID SIZE UNCHANGED BY ";
3220 PRINT "ENTERING "; C0$; "0,0"; C0$; "."
3230 PRINT
3240 PRINT "YOUR NEW DIMENSIONS";
3250 INPUT G3,G4
3260 IF G3<>0 THEN 3280
3270 IF G4=0 THEN 3410
3280 IF G3/2<>INT(G3/2) THEN 3490
3290 IF G4/2<>INT(G4/2) THEN 3510
3300 REM
3310 REM  *******  VALID INPUT
3320 REM
3330 LET G1=G3
3340 LET G2=G4
3350 PRINT
3360 PRINT "THE NEW SIZE OF THE HURKLE'S GRID IS"; G1; "BY"; G2; "."
3370 GOTO 3440
3380 REM
3390 REM  *******  0,0 INPUT
3400 REM
3410 PRINT
3420 PRINT "THE HURKLE'S GRID WILL REMAIN ITS CURRENT SIZE OF"; G1;
3430 PRINT "BY"; G2; "."
3440 PRINT
3450 GOTO 1870
3460 REM
3470 REM  *******  NON-INTEGER INPUT
3480 REM
3490 PRINT "   YOUR FIRST";
3500 GOTO 3520
3510 PRINT "   YOUR SECOND";
3520 PRINT " DIMENSION IS NOT AN EVEN INTEGER!  PLEASE TRY AGAIN..."
3530 GOTO 3230
3540 REM
3550 REM    * * * * * * *   T H E   " T R I E S "   O P T I O N
3560 REM
3570 GOSUB 3820
3580 PRINT "YOU ARE NOW ALLOWED"; N; "TRIES TO FIND THE HURKLE.  ";
3590 PRINT "ENTER YOUR NEW"
3600 PRINT "LIMIT BELOW.  YOU MAY LEAVE THE LIMIT UNCHANGED BY ";
3610 PRINT "ENTERING "; C0$; "0"; C0$; "."
3620 PRINT
3630 PRINT "YOUR NEW LIMIT";
3640 INPUT N1
3650 PRINT
3660 IF N1<>0 THEN 3690
3670 PRINT "THE NUMBER OF TRIES ALLOWED WILL REMAIN AT"; N; "."
3680 GOTO 3710
3690 LET N=N1
3700 PRINT "YOU WILL NOW BE ALLOWED"; N; "TRIES TO FIND THE HURKLE."
3710 PRINT
3720 GOTO 1870
3730 REM
```

```
3750 REM
3760 REM    * * * * * * *    S U B R O U T I N E S
3770 REM
3780 REM
3790 REM
3800 REM  *******  SCREEN CLEARER
3810 REM
3820 PRINT \ PRINT
3830 RETURN
3840 REM
3850 REM  *******  "YES", "NO", AND "QUIT" RESPONSE DECODER
3860 REM
3870 INPUT A$
3880 PRINT
3890 IF POS(A$,"Y",1)<>0 THEN 3990
3900 REM -- FOR DECSYSTEM-10, REPLACE ABOVE STATEMENT WITH:
3910 REM         IF INSTR(1,A$,"Y")<>0 THEN 3990
3920 IF POS(A$,"N",1)<>0 THEN 4010
3930 REM -- FOR DECSYSTEM-10, REPLACE ABOVE STATEMENT WITH:
3940 REM         IF INSTR(1,A$,"N")<>0 THEN 4010
3950 IF A$="QUIT" THEN 4540
3960 PRINT "     PLEASE INPUT "; C0$; "YES"; C1$; "NO"; C0$; ", OR "
3970 PRINT C0$; "QUIT"; C0$; ".  YOUR CHOICE";
3980 GOTO 3870
3990 LET A=1
4000 RETURN
4010 LET A=0
4020 RETURN
4030 REM
4040 REM    * * * * * * *    O P T I O N    P R I N T E R
4050 REM
4060 PRINT C0$; "GO"; C1$; "HELP"; C1$; "INSTR"; C1$; "QUIT"; C1$;
4070 PRINT "SIZE"; C0$;
4080 IF P0=2 THEN 4120
4090 PRINT ","
4100 PRINT "AND "; C0$; "TRIES"; C0$; ".  ";
4110 RETURN
4120 PRINT ", OR "; C0$; "TRIES"; C0$; "."
4130 RETURN
4140 REM
4150 REM    * * * * * * *    I N S T R U C T I O N S
4160 REM
4170 GOSUB 3820
4180 PRINT "A HURKLE IS HIDING IN A CARTESIAN COORDINATE GRID"
4190 PRINT "LIKE THE ONE AT THE RIGHT.  GUESS ITS LOCATION BY";
4200 PRINT "          (POS)"
4210 PRINT "ENTERING A HORIZONTAL COORDINATE FOLLOWED BY A   ";
4220 PRINT "               N"
4230 PRINT "VERTICAL ONE.  FOR EXAMPLE, THE  *  IS AT  -4,1 .";
4240 PRINT "            *  !"
4250 PRINT "POINT  0,0  IS AT THE CENTER OF THE GRID, WHERE  ";
4260 PRINT "       W  <--+--> E"
4270 PRINT "THE  *  IS   AFTER EACH GUESS, I WILL TELL YOU   ";
4280 PRINT "     (NEG)   !   (POS)"
4290 PRINT "WHERE TO GO TO FIND THE HURKLE BY SAYING "; C0$; "NORTH";
4300 PRINT C0$; "                S"
4310 PRINT "FOR THE POSITIVE VERTICAL DIRECTION, "; C0$; "WEST"; C0$;
4320 PRINT " FOR                (NEG)"
4330 PRINT "THE NEGATIVE HORIZONTAL, ETC.  GOOD LUCK!"
4340 RETURN
4350 REM
4360 REM    * * * * * * *    D I R E C T I O N A L    H I N T E R
4370 REM
4380 PRINT "GO ";
4390 IF V=V THEN 4440
4400 IF V>V THEN 4430
4410 PRINT "NORTH";
4420 GOTO 4440
4430 PRINT "SOUTH";
4440 IF H=X THEN 4490
4450 IF H>X THEN 4480
4460 PRINT "EAST";
4470 GOTO 4490
4480 PRINT "WEST";
4490 PRINT "..."
4500 RETURN
4510 REM
4520 REM    * * * * * * *    Q U I T T I N G    R O U T I N E
4530 REM
4540 PRINT
4550 IF I1<2 THEN 4650
4560 PRINT "YOU PLAYED A TOTAL OF"; I1; "GAMES AND FOUND THE ";
4570 PRINT "HURKLE IN"; I2; "OF THEM."
4580 PRINT "THAT'S A WINNING PERCENTAGE OF"; 100*I2/I1; "% !"
4590 PRINT
4600 IF I2<2 THEN 4650
4610 PRINT "IN THE"; I2; "GAMES YOU WON, IT TOOK YOU AN AVERAGE OF";
4620 PRINT I3/I2; "GUESSES TO"
4630 PRINT "FIND THE HURKLE."
4640 PRINT
4650 PRINT "BYE!"
4660 PRINT
4670 END

READY
```